Quick and painless intro to the Perl programming language

Alex-P. Natsios
drakevr@2f30.org

GreekLUG

07 Jun 2015

The Basics

Introduction

- What is Perl?
- What is perl?
- What is PERL?
- What is Perl's history?
- What does it look like?
- The motto
- Why should I use Perl?
- Why shouldn't I use Perl?

What is Perl?

- A high level modern programming language
- Open Source and Free
- General purpose
- Interpreted
- Dynamic

What is perl?

- The language Compiler/Interpreter
- Compiles and Interprets code on the fly
- can be used for simple oneliners

What is PERL?

WRONG! This is not a canonical or accepted spelling/case and should never be used!

What is Perl's history

- First version released in 1987
- Current versions include both Perl 5 and Perl 6
- First 5.x version released in 1994
- First version for Perl 6 is expected to arrive this year

What does it look like?

```
#!/usr/bin/perl
$|--;for($/=1/10;$/<=1/2;$/+=50e-7){$\=qq/J,\r/;
substr$\,$/*length$\,$/-$/,$_,for split/(.{2})/,
q/ursetk caanho tlhreerP /;print'';}warn qq/\n/;

#  by choroba from PerlMonks.org
```

What does it look like?

- kidding, that was obfuscated
- written like that for compactness, obscurity or just fun.
- normal Perl is not written like that.

What does it look like?

```
#!/usr/bin/env perl

use Lingua::Romana::Perligata;

maximum inquementum tum biguttam egresso scribe.
meo maximo vestibulo perlegamentum da.
da duo tum maximum conscribementa meis listis.

dum listis decapitamentum damentum nexto
    fac sic
       nextum tum novumversum scribe egresso.
       lista sic hoc recidementum nextum cis vannementa da listis
    cis.
```

What does it look like?

- kidding, that was just a module
- written like that for pure fun.
- has a paper to accompany and explain it though!
- normal Perl is not written like that.

What does it look like?

```
#!/usr/bin/perl -w                    # camel code
use strict;
                                          $_='ev
                                       al("seek\040D
           ATA,0,                    0;");foreach(1..3)
       {<DATA>;}my                  @camel1hump;my$camel;
  my$Camel  ;while(                 <DATA>){$_=sprintf("%-6
9s",$_);my@dromedary               1=split(//);if(defined($
_=<DATA>)){@camel1hum             p=split(//);}while(@dromeda
 ry1){my$camel1hump=0            ;my$CAMEL=3;if(defined($_=shif
        t(@dromedary1          ))&&/\S/){$camel1hump+=1<<$CAMEL;}
       $CAMEL--;if(d         efined($_=shift(@dromedary1))&&/\S/){
      $camel1hump+=1       <<$CAMEL;}$CAMEL--;if(defined($_=shift(
     @camel1hump))&&/\S/){$camel1hump+=1<<$CAMEL;}$CAMEL--;if(
     defined($_=shift(@camel1hump))&&/\S/){$camel1hump+=1<<$CAME
     L;;}$camel.=(split(//,"\040..m‘{/J\047\134}L^7FX"))[$camel1h
      ump];}$camel.="\n";}@camel1hump=split(/\n/,$camel);foreach(@
      camel1hump){chomp;$Camel=$_;y/LJF7\173\175‘\047/\061\062\063\
      064\065\066\067\070/;y/12345678/JL7F\175\173\047‘/;$_=reverse;
       print"$_\040$Camel\n";}foreach(@camel1hump){chomp;$Camel=$_;y
        /LJF7\173\175‘\047/12345678/;y/12345678/JL7F\175\173\0 47‘/;
         $_=reverse;print"\040$_$Camel\n";}’;;s/\s*//g;;eval;   eval
           ("seek\040DATA,0,0;");undef$/;$_=<DATA>;s/\s*//g;(   );;s
             ;^.*_;;;map{eval"print\"$_\"";}/.{4}/g; __DATA__   \124
               \1   50\145\040\165\163\145\040\157\1 46\040\1  41\0
                    40\143\141  \155\145\1 54\040\1   51\155\  141
                    \147\145\0  40\151\156 \040\141    \163\16 3\
                     157\143\   151\141\16  4\151\1     57\156
                     \040\167  \151\164\1   50\040\      120\1
```

Ok, What does it REALLY look like?

```
#!/usr/bin/perl
use strict;
use warnings;

use Path::Class;
use autodie; # die if problem reading or writing a file

my $dir = dir("/tmp"); # /tmp

my $file = $dir->file("file.txt");

# Read in the entire contents of a file
my $content = $file->slurp();

# openr() returns an IO::File object to read from
my $file_handle = $file->openr();

# Read one line at a time
while( my $line = $file_handle->getline() ) {
        print $line;
}
```

The motto

There's more than one way to do it

(TMTOWTDI or TIMTOWTDI, pronounced Tim Toady)

Why should I use Perl?

- General Purpose
- Fast and simple to deploy
- Fun
- Speed is not a critical issue.
- Programming freedom (TIMTOWTDI), can code however you like

Why shouldn't I use Perl?

- You have legacy code that is NOT in Perl
- Speed is a critical issue
- Need FULL optimizable performance
- Target system does not have Perl (or cannot install it)

Package and Versions Management

CPAN

Comprehensive Perl Archive Network

- Interface for installing modules
- Vast collection of Modules and Documentation
- Found at `http://www.cpan.org`

CPANMINUS

cpanm

- A self bootstrappable script
- Has NO dependencies
- Can Get, Unpack, Install CPAN modules
- Needs only 10M of RAM when running
- Zeroconf, Just works

Perl Binary Management

plenv

- Simple, UNIX like version management
- Needs git and your regular Build tool collection
  like build-essential on debian or the C/C++ development pattern on openSUSE
- excellent buddy to Carton (Perl world's "Bundler") for dependency management

The community

"One of Perl's biggest strengths is its community. . ."

Useful links

https://www.perl.org/community.html
http://www.pm.org/
http://www.perlmonks.com/

Thank you for your attention

Q & A